

CLAIMS

1. A method for testing software, comprising:
 examining an application software program
5 including calls to system classes with both a static
analysis tool and a dynamic analysis tool;
 determining a static use count of said system
classes;
 deriving a dynamic use count of each of said
10 system classes during operation of said application
software program;
 assigning a proportional weighing attribute to
each system class based on its corresponding static use
count and dynamic use count; and
15 testing said system classes in order according
to said corresponding proportional weighing attributes.
2. The method of claim 1, wherein:
 the step of testing is such that only the most
20 heavily weighted portion of all such system classes are
tested at all.
3. The method of claim 1, wherein:
 the step of testing is such that only those
25 system classes that are actually used in operation of said
application software program are tested at all;
 wherein, costs and delays associated with such
pointless testing are avoided.
- 30 4. The method of claim 1, wherein:
 producing a static use count further comprises
assigning a static observation percentage to each system
class by dividing said static use count by a sum of all
static use counts.

5. The method of claim 1, wherein:

producing a dynamic use count further comprises
assigning a dynamic observation percentage to each system
class by dividing said dynamic use count by a sum of all
dynamic use counts.

6. The method of claim 1, wherein:

producing a static use count further comprises
assigning a static observation percentage to each system
class by dividing the static use count by a sum of all
static use counts; and

producing a dynamic use count further comprises
assigning a dynamic observation percentage to each system
class by dividing the dynamic use count by a sum of all
dynamic use counts.

7. The method of claim 6, wherein the a step of
assigning to each system class a weight based on the
static use count and the dynamic use count further
comprises the steps of:

assigning to a public untested system class a
first weight defined by a first constant plus a sum of the
static use count plus the dynamic use count;

assigning a private untested software class a
second weight that is equal to the first constant;

assigning to each public function that is not
fully tested a third weight that is defines as a second
constant that is less than the first constant, to which is
added a sum of the static observation percentage plus the
dynamic observation percentage; and

assigning to all remaining public and private
functions a fourth weight defined as a third constant that
is less than the second constant.

8. The method of claim 1, wherein:

the testing the system classes further comprises ending a test when a testing resource is exhausted and prior to testing a last entry having a least weight.

5

9. The method of claim 8, wherein:

the testing the system classes further comprises ending a test when at least a limit of available time or funding is exhausted and prior to testing a last entry having a least weight.

10

10. Software for testing object-oriented system software having system classes, the software having machine readable code for performing the following steps:

15

running a static analysis tool for examining an application software program and producing a result, the application software program including calls to the system classes;

20

determining a static use count of the system classes in the application software program from the result;

25

running a dynamic analysis tool for examining the application software program and producing a dynamic use count based on the application software program's dynamic use of the system functions while running the application software program;

30

assigning to each system class a weight based on the static use count and the dynamic use count, and testing the system classes, in order, based on the assigned weight, from a first entry having a greatest weight.

11. The software of claim 10, wherein:

35

producing a static use count further comprises assigning a static observation percentage to each system class by dividing the static use count by a sum of all static use counts.

12. The software of claim 10, wherein:
producing a dynamic use count further comprises
assigning a dynamic observation percentage to each system
5 class by dividing the dynamic use count by a sum of all
dynamic use counts.

13. The software of claim 10, wherein:
producing a static use count further comprises
10 assigning a static observation percentage to each system
class by dividing the static use count by a sum of all
static use counts, and
producing a dynamic use count further comprises
assigning a dynamic observation percentage to each system
15 class by dividing the dynamic use count by a sum of all
dynamic use counts.

14. The software of claim 10, wherein the assigning
to each system class a weight based on the static use
20 count and the dynamic use count further comprises the
steps of:
assigning to a public untested system class a
first weight defined by a first constant plus a sum of the
static use count plus the dynamic use count;
25 assigning a private untested software class a
second weight that is equal to the first constant;
assigning to each public function that is not
fully tested a third weight that is defines as a second
constant that is less than the first constant, to which is
30 added a sum of the static observation percentage plus the
dynamic observation percentage; and
assigning to all remaining public and private
functions a fourth weight defined as a third constant that
is less than the second constant.

35

15. A software tester, comprising:
means for examining an application software
program including calls to system classes with both a
static analysis tool and a dynamic analysis tool;
5 means for determining a static use count of said
system classes;
means for deriving a dynamic use count of each
of said system classes during operation of said
application software program;
10 means for assigning a proportional weighing
attribute to each system class based on its corresponding
static use count and dynamic use count; and
means for testing said system classes in order
according to said corresponding proportional weighing
15 attributes.

16. The tester of claim 15, wherein:
the means for testing is such that only the most
heavily weighted portion of all such system classes are
20 tested at all.

17. A business model for testing software,
comprising:
setting a resource limit on the available time
25 or money that is devoted to testing a particular
application software program;
examining said application software program
including calls to system classes with both a static
analysis tool and a dynamic analysis tool;
30 determining a static use count of said system
classes;
deriving a dynamic use count of each of said
system classes during operation of said application
software program;
35 assigning a proportional weighing attribute to
each system class based on its corresponding static use
count and dynamic use count;

testing said system classes in order according
to said corresponding proportional weighing attributes and
proceeding down to the least heavily weighted system
classes; and

5 stopping testing when said resource limit is
reached.